**International Journal of Advanced Research in Engineering and Technology (IJARET)**
Volume 7, Issue 1, Jan-Feb 2016, pp. 07-16, Article ID: IJARET_07_01_002
Available online at
http://www.iaeme.com/IJARET/issues.asp?JType=IJARET&VType=7&IType=1
Journal Impact Factor (2016): 8.8297 (Calculated by GISI) www.jifactor.com
ISSN Print: 0976-6480 and ISSN Online: 0976-6499


# A NOVEL RESEARCH PAPER RECOMMENDATION SYSTEM

**Madhushree B**
L. J. Institute of Engineering and Technology, Ahmedabad

**ABSTRACT**

*Research often spend a considerable amount of time searching for published papers and articles relevant to their interest, dissertation and research work. A recommender engine is a tool, a means to answer the question. "What are the best recommendations for a user?" Using trust in social networks provides a promising approach to make recommendations to other user based on trust propagation in finding research papers or research papers of a friend/research with similar interests. However, current recommendation algorithms are based on user-item rating. A collaborative filtering based research paper recommender system is proposed here with User and Item Based collaborative filtering approach to implement a recommender system for Research Paper. Users will enter details when they create a profile with the system. Based on the detailed profile and current contents of the recommended item in the database, the system recommends different research paper to user. User Based Recommender uses preferences of their other similar users for recommendation of research paper. Item Based Recommender uses User's Item visit history for Recommendation. Four algorithms are implemented in each category by using Standard Dataset. Based on result generated, all methods are compared based on recommender accuracy in terms of Precision and Recall.*

**Key words:** Search Engines, Database, Data Mining

**Cite this Article:** Madhushree B, A Novel Research Paper Recommendation System. *International Journal of Advanced Research in Engineering and Technology*, **7**(1), 2016, pp. 07-16.
http://www.iaeme.com/IJARET/issues.asp?JType=IJARET&VType=7&IType=1

## 1. INTRODUCTION

Recommender systems are supposed to help users navigate through complex information spaces by suggesting which items a user should visit and which items a user should not. They have proven to be successful in many domains, including news, movies, jokes and friends among others. Even more recommenders have transitioned

from a research curiosity into products and services used every day including Amazon.com, Facebook, Yahoo!, Music, TiVo and even Apple's iTunes Music Store.

Yet, with this growing usage, there is a feeling that recommenders are not living up to their initial promise. Recommenders have mostly been applied to lower-density information spaces where users are not required to make an intensive effort to understand and process recommended information (movies, music, and jokes). Moreover, recommenders have supported a limited number of tasks.

The useful recommendation is one that meets a user's current, specific need. It is not a binary measure, but rather a concept for determining how people use a recommender, what they use one for, and why they are using one. Current systems, such as e-commerce websites, have predefined a user's need into their business agendas-they decide if a system is useful for a user. Users have their own opinions about the recommendations they receive, and behalf it is recommenders should make personalized recommendations, they shouldn't listen to user's personalized opinions.

There are many recommender pitfalls. These include not building user confidence (trust failure), not generating any recommendations (knowledge failure), generating incorrect recommendations (personalization failure), and generating recommendations to meet the wrong need (context failure), among others. Avoiding these pitfalls id difficult yet critical for the continued growth and acceptance of recommenders as knowledge tools.

The key feature of recommender systems is personalization. Unlike search engines recommenders take into account the personality and past behavior of each user. A typical recommender wouldn't present the same set to two different users.

Recommender systems are programs operating on large amount of data in software systems. Recommender systems try to present items such as books, music, news, etc. That are likely to be interesting for a given user. These systems may be helpful for users that are choosing between a large number of items and aren't willing to browse information about all available items.

Recommender Systems typically apply techniques and methodologies from other neighboring areas – such as Human Computer Interaction (HCI) or Information Retrieval (IR). However, most of these systems bear in their core an algorithm that can be understood as a particular instance of a Data Mining technique [1]. Recommender Systems will benefit the customer by making to him/her suggestions on items that he/she is assembly going to like. At the same time, the business will be benefited by the increase of sales which will normally occur when the customer is presented with more items he would likely find appealing.

The two basic entities which appear in any Recommender System are the user (sometimes also referred to as customer) and the item (also referred to as product). A user is a person who utilizes the Recommender System providing his opinion about various items and receives recommendations about new items from the system. Recommendation is based on the preferences of the recommender (and perhaps of the seeker and other individuals). A preference is an individual mental state concerning a subset of items from the universe of alternatives. Individuals form preferences based on their experience with the relevant items, such as listening to music, watching movies, tasting food, etc. Figure 1 summarizes the concepts and situates them in a general model of recommendation.

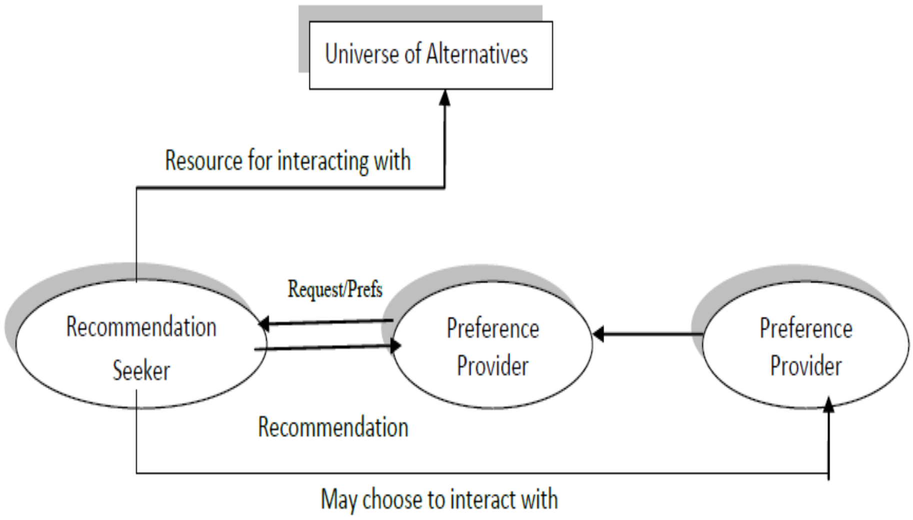


**Figure 1** Model of the Recommendation Process

For a successful implementation of a recommender system, several conditions have to be fulfilled.

1. Many items: In the domain there are many items that might be interesting for a user. It is not possible for the user to browse all of them.
2. Choice based on taste: The choice of the items depends on the taste of each user. It there were some objective criteria for recommending items to users the recommender system would not be much helpful.
3. Taste data: In the system there have to be some data about the users interacting with items that can be interpreted as expressing taste. It does not matter if these are explicit rating data or implicit capture or user behavior, but there have to be some.
4. Homogeneous items: The items in the domain have some common attributes, they can all be covered by taste data, and e.g. they can all be viewed or rated.

Recommender systems are useful for both users and system holders. Nowadays the amounts of information we are retrieving have become increasingly enormous. John Naisbitt observed that: “we are drowning in information but starved for knowledge.” This “starvation” caused by having many ways people pour data into the Internet but not many techniques to process the data to knowledge. For example, digital libraries contain tens of thousands of journals and articles. However, it is difficult for users to pick the valuable resources they want. What we really need is new technologies that can assist us find resources of interest among the overwhelming item available. One of the most successful such technologies is the Recommender system; “a personalized information filtering technology used to either predict whether a particular user will like a particular item (prediction problem) or to identify a set of N items that will be of interest to a certain user (top-N recommendation problem)” [2].

Over the years, various approaches for building recommender systems have been created; collaborative filtering has been a very successful approach in both research and practice, and in information filtering and e-commerce applications. Collaborative filtering works by creating a matrix of all items and users’ preferences. In order to recommend items for the target user, similarities between him and other users are computed based on their common taste.

## 2. LITERATURE SURVEY

The term collaborative filtering was first coined by Gold-berg to describe an email filtering system called Tapestry. Collaborative Filtering is a social environment based method of recommendation used to propose items which like-minded users favor (and the active user has not yet seen). These recommendations match a user's needs based upon information gathered over time from other people having interests matching that of the current user. This approach gives recommendations based on correlation between users. Collaborative Filtering is the pivot of modern day recommender systems. Collaborative Filtering is effective since people's tastes are typically not orthogonal. A Collaborative Filtering scheme aims to make suggestions to users based upon his/her previous likings and also the preferences of like-minded uses i.e. users falling into similar categories/groups/communities as the current user [3].

User-based recommendation finds similar users, and sees what they like. Item-based recommendation sees what the user likes, and then finds similar items. In general, collaborative filtering methods are currently the most used and the most successful methods for recommending. When properly used, they provide high accuracy and scalability for large amount of data. However they have some drawbacks too.

1. Cold start problem: As for the content-based method, the system doesn't know any liked objects for a new subject. For a user being a subject, this problem can be solved by asking the user to enter some data. This approach is used in the Netflix system. A new user is given a questionnaire for rating some chosen movies. Another solution to the problem is using a hybrid recommender that would use an easier recommender (like giving the most popular objects) when predictions from collaborative filtering aren't available.
2. Sparsity: The subject-object matrix is usually very sparse – the number of known relationships is very small compared to the number of relationships that should be predicted. Therefore, in most of the collaborative filtering methods, the objects related to few subjects are seldom recommended. This drawback is overcome by some of the model-based methods, e.g. matrix factorization.
3. Grey sheep problem: In this system, there might be subjects whose preferences aren't consistently similar to other subjects. Therefore they don't belong to any preference subject group and they can't get any accurate recommendations [4].

Amazon: King of Recommendations. Unsurprisingly, Amazon used all 3 approaches (personalized, social and item) [5]. Amazon's system is very sophisticated, but at heart all of its recommendations "are based on individual behavior, plus either the item itself or behavior of other people on Amazon." What's more, the aim of it all is to get you to add more things to your shopping cart. Other newer Internet companies have tended to focus on specific methods of recommendation.

Google: Focus on Personalized Recommendations. The most successful internet company of this era has without a doubt been Google. It too has been using recommendation technologies to improve its core search product [6]

There are two ways that George does this:

1. Google customizes search results "when possible" based on our location and/or recent search activity.
2. When were signed in to your Google Account, we "may see even more relevant useful results based on your web history."

So Google is using both our location and our personal search history to make its search results supposedly stronger. This is very much the ‘personalized recommendation’ approach – and indeed personalization has been a buzzword for Google in recent years. However, the two other types of recommendation are also present in Google’s core search product:

1. Google’s search algorithm PageRank is basically dependent on social recommendations – ie. Who links to a webpage?
2. Google also does item recommendations with its “Did you mean” feature.

There are surely other ways recommendations technologies are being deployed in Google search – not to mention the range of other products Google has. Google News, its start page iGoogle, and its commerce site Froogle all have recommendation features.

You Tube: Uses Amazon’s Algorithm for Recommendation Engine. This is an interesting move being that Google has the man power and smarts to build a fairly good recommendation on their own. But here they opt to use an algorithm designed by Amazon. Of course, the best algorithms are enhancements on top of previous algorithms. Google’s own algorithm is light-years beyond where they first were with their original PageRank patent. Recommending interesting and personally relevant videos to YouTube users is a unique challenge: Videos as they are uploaded by users often have no or very poor metadata. The video corpus size is roughly on the same order or magnitude as the number of active users. Furthermore, videos on YouTube are mostly short form (under10 minutes in length). User interactions are thus relatively short and noisy unlike Netflix or Amazon where renting a movie or purchasing an item are very clear declarations of intent. In addition, many of the interesting videos on YouTube have a short life cycle going from upload to viral in the order of days requiring constant freshness of recommendation.

To compute personalized recommendations YouTube combine the related videos association rules with a user’s personal activity on the site: This can include both videos that were watched (potentially beyond a certain threshold), as well as videos that were explicitly favorited, “liked”, rated, or added to playlists. Recommendations are the related videos, for each video the user has watched or liked after they are ranked by video quality, user’s unique taste and preferences and filtered to further increase diversity. To evaluate recommendation quality they use a combination of different metrics. The primary metrics they consider include click through rate (CTR), long CTR (only counting clicks that led to watches of a substantial fraction of the video), session length, time until first long watch, and recommendation coverage (the fraction of logged in users with recommendations). They use these metrics to both track performance of the system at an ongoing basis as well as for evaluating system changes on live traffic [7]

## 3. PROPOSED APPROACH

One approach to the design of recommender systems that has seen wide use is collaborative filtering. Collaborative filtering methods are based on collecting and analyzing a large amount of information on users’ behaviors, activities or preferences and predicting what users will like based on their similarity to other users. User-based collaborative filtering attempts to model the social process of asking a friend for a recommendation. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an “understanding”

of the item itself. One of the most famous examples of Collaborative Filtering is item-to-item collaborative filtering (people who buy x also buy y), an algorithm popularized by Amazon's recommender system.

User based algorithms are CF algorithms that work on the assumption that each user belongs to a group of similar behaving users. The basis for the recommendation is composed by items that are liked by users. Items are recommended based on users' tastes (in term of their preference on items). The algorithm considers that users who are similar (have similar attributes) will be interested on same items [8]. User based algorithms are three steps algorithm;

For every item I that u has no preference for yet

For every other user v that has a preference for i

Compute a similarity s between u and v

Incorporate v's preference for I, weighted by s, into a running average

Return the top items, ranked by weighted average

1. Profile every user in order to find which ones are similar to the target user.
2. Compute the union of the items selected by these users and associate a weight with each item based o its importance in the set and
3. Select and recommend items that have the highest weight and have not been already selected by the active user

The most important step is the first one; creating the union of items liked by others or selecting the most important of them is easily done when the set of similar users is known. Thus the overall performance of the algorithm will depend on the method used to find users that are similar to the target user. There are many methods by which it can be done. The k-Nearest Neighbors algorithm is the most used because of its efficiency

It would be terribly slow to examine every item. In reality, a neighborhood of most similar users is computed first, and only items known to those users are considered:

for every other user w

Compute a similarity s between u and w

Retain the top users, ranked by similarity, as a neighborhood n

for every item I that some user in n has a preference for,

but that u has no preference for yet

for every other user v in n that has a preference for i

compute a similarity s between u and v

incorporate v's preference for I, weighted by s, into a running average

The primary difference is that similar users are found first, before seeing what those most-similar users are interested in. Those items become the candidates for recommendation. The rest is the same. This is the standard user-based recommender algorithm.

User-Based Recommender System in Mahout

DataModel model = new FileDataModel (new File ("intro.csv"));

userSimilarity similarity = new PearsonCorrelationSimilarity (model);

UserNeighborhood neighborhood =

New NearestNUserNeighborhood (2, similarity, model);

Recommender recommender =

New GenericUserBasedRecommender (model, neighborhood, similarity);

User Similarity encapsulates some notion of similarity among users, and User-Neighborhood encapsulates some notion of a group of most-similar users. These are necessary components of the standard user-based recommender algorithm.

Item-based recommender are a type of collaboration filtering (CF) algorithms that look at the similarity between items to make a prediction. The idea is that a user is most likely to purchase items that are similar to the one he already bought in the past; so by analyzing the purchasing information we can have an idea about what the may want in the future (Deshpande, Karpis 2004). Analyzing the historical information can be done explicitly (by looking at the explicit ratings users made on the items) or implicitly (for example through the user browsing information or the rating on categories of item).

Item-based algorithms are two steps algorithms:

1. Scan the past information of the users; the ratings they gave to items are collected during this step. From these ratings, similarities between items are built and inserted into an item-to-item matrix M. The element xi; j of the matrix M represents the similarity between the item in row I and the item in column j.
2. Selects items that are most similar to the particular item a user is rating.

for’ every item I that u has no preference for yet

For every item j that u has a preference for

Compute a similarity s between I and j

Return the top items, ranked by weighted average

Pearson Correlation Similarity still works here because it also implements the Item Similarity interface, which is entirely analogous to the User Similarity interface. It implements the same notion of similarity, based on the Pearson correlation, but between items instead of users. That is, it compares series of preferences expressed by many users, for one item, rather than by one user for many items. Generic Item Based Recommender is simpler. It only needs a Data Model and Item-Similarity-there’s no Item Neighborhood.

## 4. IMPLEMENTATION

A recommender that could predict all our preferences exactly would present all items ranked by us future preference and he done. These would be the best possible recommendations. Training data and scoring nobody knows for sure how we’ll like some new item in the future. In the context of a recommender engine, this can be simulated by setting aside a small part of the real data set as test data. These test preferences aren’t present in the training data fed into a recommender engine under evaluation. Instead, the recommender is asked to estimate preferences for the missing test data, and estimates are compared to the actual values.

Evaluating precision and recall: Precision and Recall are two popular decision support metrics from information retrieval. In recommender systems, they judge relevance by counting recommendation ‘hits’: if a withheld item appears in a top-n recommendation list. These measures are also done per user and averaged over all users. For one user, recall measures how many of the withheld items appeared in the recommendation list:

Precision = Correctly recommended items/ total recommended items = d/b+d

Precision, on the other hand, measures how many of the recommended items appear in the list of withheld items.

Recall= Correctly recommended items/ total recommended items = d/c+d

These two measures complement each other's by measuring the relevance from both the point of view of the items being recommended and the items known to be good.

The proposed algorithms should implements fall under the broad umbrella of machine learning or collective intelligence.

The quality of recommendations is largely determined by the quantity and quality of data. Recommender algorithms are data-intensive by nature; their computations access a great deal of information. Runtime performance is therefore greatly affected by the quantity of data and its representation. Intelligently choosing data structures can affect performance by orders of magnitude, and, at scale, it matters a lot. The abstraction encapsulates recommender input data.

The input to a recommender engine is preference data – who likes what, and how much. That means the input to the recommender is simply a set of user ID, item ID, and preference value tuples – large set, of course. Sometimes, even preference values are omitted. A preference is the most basic abstraction, representing a single user ID, item ID, and preference value. One object represents one user's preference for one item.

Another important part of user-based recommenders is the User Similarity implementation. A user-based recommender relies most of all on this component. Without a reliable and effective notion of which users are similar to others, this approach falls apart. Various similarity models are used based on types of Dataset. Separate similarity models are explored for dataset with item preferences, without item preferences and Boolean preferences.

A neighborhood of the n most similar users, but rather try to pick the pretty similar users and ignore everyone else. It's possible to pick a similarity threshold and take any users that are at least that similar. Figure 2 illustrates a threshold-based definition of user neighborhood.

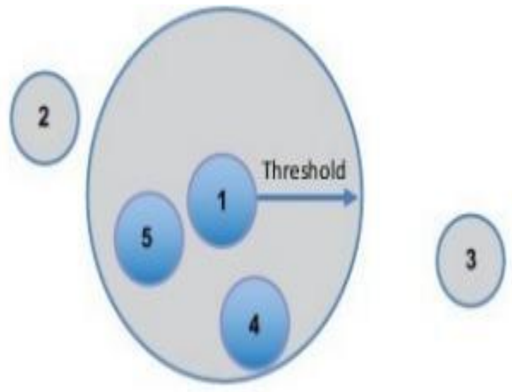


**Figure 2** Defining a neighborhood of most similar users with a similarity threshold

The threshold should be between -1 and 1, because all similarity metrics return similarity values in this range. The notation used for the system is:

UserNeighborhood neighborhood = New ThresholdUserNeighborhood (0.1, userSimilarity, model);

A recommender engine is a tool, a means to answer the question, "What are the best recommendations for a user?" A recommender that could predict all our preferences exactly would present all items ranked by us future preference and be done. These would be the best possible recommendations.

To generate an aggregated score, we use for the example a weight of 1 for all users. Thus the ones in the selected users are just summed up. Then items known to the active user are removed and the N items with the highest score (greater than zero) form the top-N recommendations. In the example in Figure 7 only two items are recommended.

The top-10 recommendation items recommended by each algorithm are described by User Based Recommender and Item Based Recommender on above datasets. Based on the general recommendations all similarity models are compared based on Precision and Recall

Here, also recommender accuracy is higher when we consider the User Based Recommender for the dataset where only presence of Paper preferences are consider and the explicate value of paper preferences are impractical. But here LogLike Similarity is better than Tanimoto Similarity.

It is clear that the recommendations are more accurate then preferences are not considered. Both LogLike and Tanimoto Similarities are superior than similarities with Paper preferences are considered. Based on Precision and recall values the Recommender with Tanimoto Coeffiecient Similarity is good choice for recommending research Papers.

## 5. CONCLUSION AND FUTURE WORK

Recommender system have developed in response to a manifest need: helping people deal with the world of information abundance and overload. Further it has become clear that they can link people with other people who share their interests, not just with relevant information. A set of four major issues for recommender systems were identified: how preferences data is obtained and used, the roles played by people and by computation, and the types of communication involved, algorithms for linking people and computing recommendations, and presentation of recommendations to users. Then it is identified four major approaches to recommender systems, which can be distinguished in large part by which of the issues they address, and how they address them. For research paper domain where only presence of paper preferences are important rather than content of the research paper or their explicate ratings. There is a number of directions for future work, including evaluating the methods, exploring indexing methods to speed up.

## REFERENCES


[1] A. J. N. O. a. J. M. P. Xavier Amatriain, Data Mining Methods for Recommender, Springer Science+Business Media, vol. 10, no. 7, pp. 39-71, 2011.

[2] M. D. a. G. KARYPIS, Item-Based Top-N Recommendation, ACM Transactions on Information Systems, vol. 22, no. 1, p. 143–177, January 2014.

[3] I. a. H. P. Yakut, Privacy-Preserving Hybrid Collaborative Filtering On Cross Distributed Data, Knowledge and Information Systems, vol. 30, no. 2, pp. 405-433, 2011.

[4] Isak Shabani and Amir Kovaçi, Communication between Distributed Systems Using Google Infrastructure. *International Journal of Computer Engineering and Technology*, **4**(6), 2014, pp. 386-393.

[5] Mark Claypool, Anuja Gokhale, Tim Mir, Pavel Murnikov, Dmitry Netes, and Matthew Sartin, Combining content-based and collaborative filters in an online newspaper, In Proceedings of ACM SIGIR Workshop on Recommender Systems, 1999.

[6] B. S. a. J. Y. Greg Linden, Amazon.com recommendations Item-to-Item Collaborative Filtering, Published by the IEEE Computer Society, vol. 03, no. February, pp. 76-80, 2003.

[7] Jiahui Liu, Peter Dolan, Elin Rønby Pedersen, Personalized News Recommendation Based on Click, Research at Google, pp. 1-10, 2014.

[8] R. P. A. Patil, Study of Collaborative Filtering Recommendation Algorithm - Scalability Issue, International Journal of Computer Applications, vol. 61, no. 25, pp. 10-15, 013.

[9] S. K. L. G. Renjie Zhou, the Impact of YouTube Recommendation System on Video View," in IMC'10, Melbourne, Australia, 2010.

[10] Bindu Priya, Dr. N. Chandra Sekhar Reddy, Dr. Poshal and Ramya Krishna, Implementation of Geo-Messaging Using Google Cloud Messaging and Location Based Services Api of Android. *International Journal of Computer Engineering and Technology*, **5**(1), 2015, pp. 62-67.

## AUTHOR PROFILE

**MADHUSHREE B** is an Assistant Professor in L. J. Institute of Technology, Ahmedabad. She has completed her B.E. and M.E. degrees from Bangalore University. Her areas of interest include Computer Networks and Machine learning.